

XML Version of the ASDI Feed:

Interface Control Document

Version 0.4

10 March 2006

DRAFT

Prepared by

Volpe Center
55 Broadway
Cambridge, MA 02142

Table of Contents

1.	Introduction.....	3
1.1.	Background.....	3
1.2.	Purpose of this Document.....	3
1.3.	Contacts.....	3
1.4.	References.....	3
2.	Overview of the XML Data.....	4
2.1.	Compression	4
2.2.	Header	4
2.3.	Heartbeat Messages.....	6
2.4.	XML Data Format Ground Rules	6
2.5.	How to Read the Tables that Describe the Data.....	7
3.	XML Version of the ASDI Messages	9
3.1.	ASDI_DATA Tags	10
3.2.	AF Message	11
3.3.	AZ Message	13
3.4.	BZ Message	14
3.5.	DZ Message	15
3.6.	FZ Message.....	16
3.7.	RT Message	17
3.8.	RZ Message	19
3.9.	TO Message	20
3.10.	TZ Message.....	21
3.11.	UZ Message	22
	Appendix A: Sample XML Data	23
	Appendix B: Issues Concerning Compression.....	28

Revision History			
Version	Date	Author's Initials	Description of Change
0.1	3/6/06	MH & RO	Initial draft.
0.2	3/9/06	MH & RO	Internal edits.
0.3	3/9/06	MH & RO	Internal edits.
0.4	3/10/06	RO	Internal edits.

1. Introduction

1.1. Background

Since the 1990s the FAA has provided industry with a feed of flight data called the Aircraft Situation Display to Industry (ASDI) feed. This feed consists of messages that are in ASCII text, except for the RT message, which is in an archaic format that includes ASCII and encoded binary data. For documentation on the current feed, see reference 1 in section 1.4. The FAA has decided to convert this feed to XML, which has become a standard format for data exchange.

1.2. Purpose of this Document

The purpose of this document is to provide a draft of the message format that is proposed. Industry is invited to comment on this draft.

An interface control document (ICD) in general has two main components.

- It tells how an application would need to be written to access the ASDI data. That is, it describes what a program needs to do to obtain the data.
- It tells how the data is formatted. That is, it tells what a program needs to do to interpret the data.

This document concentrates on the second of these components since the new feed will be accessed in exactly the same way as the old feed. That is, a client will open a TCP/IP socket, send in a client name, and send in a client password; once these three steps have been successfully completed, the server will start streaming data to the client. Since this is documented on page 9 of reference 1, no further discussion is needed here.

1.3. Contacts

If you have questions or comments on this document, contact Rick Oiesen, Volpe Center, (617) 494-2309, oiesen@volpe.dot.gov.

If you have questions about becoming an ASDI vendor, contact Matt Maki, Volpe Center, (617) 494-3948.

If you have questions or comments on FAA management of the ASDI program, contact Judy Morrill, judy.morrill@faa.gov, (703) 326-3909.

1.4. References

1. “Aircraft Situation Display to Industry: Functional Description and Interface Control Document,” Volpe Center, report no. ASDI-FD-001, version 5.4, 15 November 2005. This document and the current version of all other ASDI documentation can be downloaded from

<http://www.fly.faa.gov/ASDI/asdi.html>

2. Overview of the XML Data

2.1. Compression

Aside from using XML instead of ASCII, there is one other significant change in the XML version of the ASDI feed. This change is that compression will be used. The reason for using compression is that, while XML is easy to read, it achieves this by using tags that greatly increase the amount of data that needs to be transmitted. If compression were not used, the bandwidth required to carry the ASDI feed would at least triple, which is not desirable.

The proposed method to deal with compression is the following.

- When the ASDI Server receives a message, it will hold it until either m messages have arrived or s seconds have elapsed, at which time it will send the accumulated messages. m and s are configurable parameters.
- When it is time to send the accumulated messages, the ASDI Server will convert them to XML format and compress the converted data using gzip.
- The ASDI Server will prepend a header of fixed size to the zipped data to provide the information needed to read and unzip it. (See the next section for discussion of this header.)
- The ASDI Server will transmit the header and zipped data.
- The client will read the fixed header and determine the size of the zipped data.
- The client will then read the entire zipped data packet and unzip it.
- The client processes these unzipped XML messages.

An important question is: **What values should be selected for m and s ?** Preliminary work indicates that if we set $m = 12$, then the bandwidth used by the XML feed would be roughly the same as the bandwidth used by the current feed. If we set m to a higher number, the required bandwidth would be reduced. Keep in mind that during a busy period there can be an average of more than a hundred messages per second for a period of hours.

One possibility would be to set $m = 64$ and $s = 1$. During a busy period, this would, compared to the current feed, reduce bandwidth usage by about a third, and no message would be delayed by more than a second. Raising m above 64 produces only small improvements in the compression factor. The argument against setting m this high is that even though the total amount of data transmitted is smaller, it is lumpier, and this could cause more of a communications backlog.

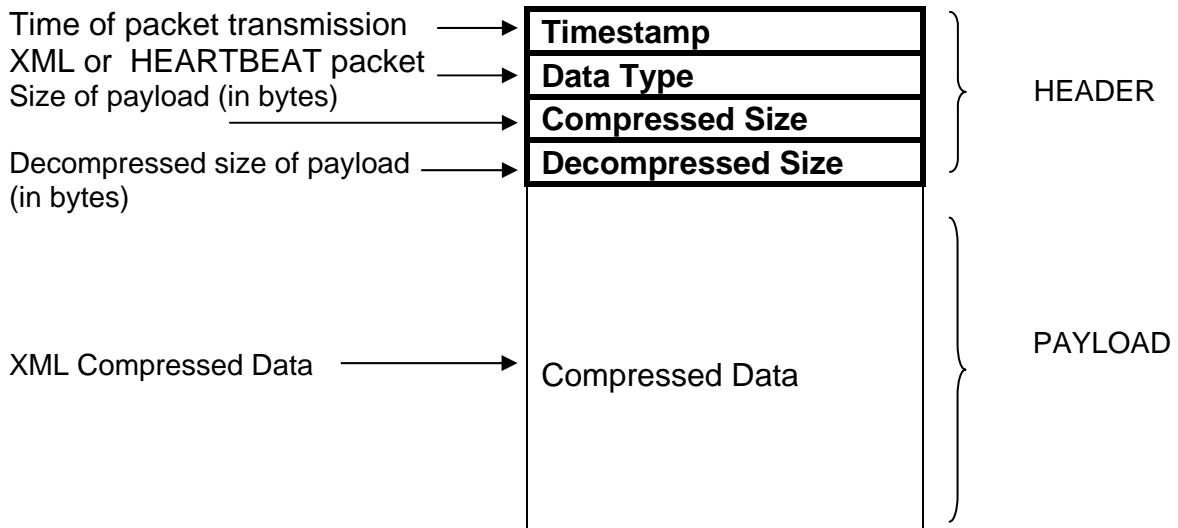
One should keep in mind that if we set $s = 1$ and if, say, only one message is received before the timeout interval has elapsed, this means that the bandwidth is being used inefficiently, but this is not a problem since bandwidth is available during this slack time.

Appendix B describes analysis that backs up some of the discussion above. Vendor input on the question of how to set m and s is encouraged.

2.2. Header

The format of the header that goes at the beginning of a batch of messages is as follows.

Figure 2-1: Format of a Transmission



The timestamp header component will be a 14 character ASCII timestamp in yyyyymmddhhmmss format that shows when the packet was transmitted by the ASDI server.

It is proposed that all other components of the data header be binary data, represented by 4-byte integer values that would be converted to network format before being sent to the vendors. This means that this header is always the same size, namely 26 bytes, so it can always be read unambiguously.

One drawback to the design described here is that the compressed size and decompressed size fields in the header are not human-readable since they are in binary. An alternative would be to express them in ASCII in fields of fixed length with padding added as needed. The trade-off between these two approaches is human-readability versus a cleaner design. Usually we assign a high value to human-readability, but the value in this case is reduced since the bulk of the message, being compressed, is not human-readable. From the Volpe point of view, either approach is acceptable. We solicit vendor feedback on this issue.

The following data structure, written in C, describes the proposed components of this header.

```
struct typedef xml_header_t
{
    char timestamp[14];
    int data_type;
    int compressed_size;
    int decompressed_size;
} xml_header_t;
```

As mentioned, this structure will be converted to network format before it is sent to the ASDI vendors. Upon receipt, the ASDI vendor will need to unpack the data and convert it back to host format. The following is a sample of C code that could be used to perform this conversion. This sample presumes that the data has already been read into `buff` from the data stream.

```

format_header(char *buff, xml_header_t *header)
{
    char *ptr;

    /* Unpack data stream */
    ptr = buff;
    memcpy(header->timestamp, ptr, sizeof(header->timestamp));
    ptr+= sizeof(header->timestamp);
    memcpy(header->data_type, sizeof(header->data_type));
    ptr+= sizeof(header->data_type);
    memcpy(header->compressed_size, sizeof(header->compressed_size));
    ptr+= sizeof(header->compressed_size);
    memcpy(header->decompressed_size, sizeof(header->decompressed_size));

    /* Convert to host format */
    header->data_type = ntohs(header->data_type);
    header->compressed_size = ntohs(header->compressed_size);
    header->decompressed_size = ntohs(header->decompressed_size);
}

```

2.3. Heartbeat Messages

In the current ASDI feed, a HEARTBEAT message is periodically sent to vendors to indicate that their connection to the ASDI server is intact. If there is a data outage due to the fact that ASDI is not receiving data from NAS.DIST, then this HEARTBEAT message assures vendors that the connection is active, but no data is available.

In the XML feed we are proposing that this information be transmitted in a header with a unique data type indicating that it is a heartbeat rather than XML data packet. 0 will indicate that a data packet is a heartbeat message; 1 indicates that the message contains data. The 'HEARTBEAT' packet would consist of a header only, with the compressed and uncompressed size elements set to 0. ASDI would send this packet periodically to indicate that the connection is still active.

2.4. XML Data Format Ground Rules

The data is in a subset of XML, with which the reader is assumed to have some familiarity. The following are the basic rules for the format of the decompressed XML data.

- The data contains only printable ASCII characters.
- The data format follows XML structural conventions.
- In XML terminology, the data is guaranteed to be "well-formed." This means that every opening tag <TAG> has a corresponding closing tag </TAG>, opening and closing tag pairs are correctly matched and nested, and consistent capitalization is used.
- The data is guaranteed to be "valid", meaning that the content matches the current documentation (this document).
- Only a simple subset of XML is used. All content is between matching start and end tags:

$$<\text{TAG}>\text{content}</\text{TAG}>$$
- The data is only in the XML format specified in this document. For example, composite tags (e.g., <TAG="content" />) and attributes are never used.

- A pair of matching start and end tags, together with the data between them, is known as a *data element* or simply an *element*.
- The data has *structure*, i.e., elements can contain other elements. An element that contains other elements is referred to as a *container*. The container element is considered to be the parent to the elements contained within. Example:

```
<HEADER>
  <SEQ>0001</SEQ>
  <TIMESTAMP>13145939</TIMESTAMP>
  <SRC>CCZW</SRC>
</HEADER>
```

- Characters that are not between matching start and end tags are to be ignored, and may be used occasionally for comments or to improve readability. Example:

```
<TAG1>
  <TAG2>This is content </TAG2>
  This is a comment.
  <TAG2>This is content</TAG2>
</TAG1>
```

- New-line characters between matching start and end tags are part of the element's data.
- Elements can be in any order within their parent element's tag pair (if there is one) or within the data .

2.5. How to Read the Tables that Describe the Data

For each message there is in the next section a table that describes the data file format. The developer should be aware of a couple of features of the data file format.

- The first line of every file is the standard "<?XML . . . >" tag, identifying the XML version number. There is no corresponding end tag.
- The table is organized according to the hierarchy of the data. That is, each tag that defines a container is followed by the elements that belong in that container. This nesting is explicitly described in the **Parent** column of each table, defined below.

The table for each data type has the following columns.

- **Tag** – The name of the tag as it appears in the file.
- **Lv** – The nesting depth of the element in the file structure. Example: ASDI_DATA is at level 1, the highest level. MSG is at level 2, because it appears within the ASDI_DATA tag pair.
- **#** – The number of these elements that can be present within the parent. For example, there can be only one HEADER in each MSG. Yet, there can be many MSG messages within a ASDI_DATA.
- **Rq?** – Describes whether, for each message type, the element is required, optional etc.:
 - **Y** Required.
 - **N** Optional.
 - **-Y-** This element must be present if its parent is present.
 - ***Y*** One and only one of these elements must be present in the parent container
 - **#Y#** At least one of these child elements must be present in the parent container

- **Parent** – The container element inside which this element appears. For example, in this example, <HEADER> is the parent of <SEQ>.
- **Example** – Provides an example of the element
- **Description** – Describes the element. If appropriate, all the allowed data values are listed and CAPITALIZED.

3. XML Version of the ASDI Messages

The following tables describe the proposed XML format for each message type in the current ASDI feed. As illustrated in the sample data in Appendix A, each XML ASDI data file will have a parent level tag of <ASDI_DATA> for each collection of <MSG> container records that are bundled together for transmission to ASDI clients. The ASDI_DATA container follows. In the subsequent descriptions of the ASDI messages, note that each is describing a single instance of a message (thus the ‘#’ field is set to 1). In the actual feed, there will be multiple <MSG> containers in single <ASDI_DATA> container.

3.1. ASDI_DATA Tags

Tag	Lv	#	Rq?	Parent	Example	Description
ASDI_DATA	1	1	Y	NONE	<ASDI_DATA>asdi data</ASDI_DATA>	Container for collection of <MSG> containers

3.2. AF Message

Tag	Lv	#	Rq?	Parent	Example	Description
MSG	2	1	Y	ASDI_DATA	<MSG>data</MSG>	Container for message
HEADER	3	1	Y	MSG	<HEADER>Header data</HEADER>	Container for header data
SEQ	4	1	Y	HEADER	<SEQ>0012</SEQ>	Sequence number of message. Valid values are 0000 – FFFF (Hex)
TIMESTAMP	4	1	Y	HEADER	<TIMESTAMP>10134248</TIMESTAMP>	Timestamp when AF message was sent by host (ddhhmmss)
SRC	4	1	Y	HEADER	<SRC>KZAU</SRC>	Center or Tracon in which AF generated
TYPE	3	1	Y	MSG	<TYPE>AF</MSG>	Type of message - AF
ID	3	1	Y	MSG	<ID>Flight ID data</ID>	Container for Flight ID data
ACID	4	1	Y	ID	<ACID>SWA917</ACID>	Flight call sign
CID	4	1	N	ID	<CID>123</CID>	Computer ID of flight
ORIG	3	1	Y	MSG	<ORIG>MDW</ORIG>	Origin
DEST	3	1	Y	MSG	<DEST>ISP</DEST>	Destination
AMENDED_DATA	3	1	Y	MSG	<AMENDED_DATA> data</AMENDED_DATA>	Container for Amendment data
AMND_AIRCRAFT_DATA	4	1	#Y#	AMENDED_DATA	<AMND_AIRCRAFT_DATA>Aircraft data </E_TYPE>	Container for Amended Aircraft DATA
AMND_NUM_ACFT	5	1	#Y#	AMND_AIRCRAFT_DATA	<AMND_NUM_ACFT>2</AMND_NUM_ACFT>	Number of aircraft
AMND_SPECIAL_ACFT	5	1	#Y#	AMND_AIRCRAFT_DATA	<AMND_SPECIAL_ACFT>L </AMND_SPECIAL_ACFT>	Special Aircraft Qualifier
AMND_EQUIP_TYPE	5	1	#Y#	AMND_AIRCRAFT_DATA	<AMND_EQUIP_TYPE>A319 </AMND_EQUIP_TYPE>	Aircraft Type
AMND_AIRBORNE_EQUIP_QUAL	5	1	#Y#	AMND_AIRCRAFT_DATA	<AMND_AIRBORNE_EQUIP_QUAL>Q </AMND_AIRBORNE_EQUIP_QUAL>	Airborne Equipment Qualifier
AMND_SPEED	4	1	#Y#	AMENDED_DATA	<AMND_SPEED>418</AMND_SPEED>	Amended Speed
AMND_COORDINATION_FIX	4	1	#Y#	AMENDED_DATA	<AMND_COORDINATION_FIX>KEEHO </AMND_COORDINATION_FIX>	Amended Coordination Fix
AMND_COORDINATION_TIME	4	1	#Y#	AMENDED_DATA	<AMND_COORDINATION_TIME>P1410 </AMND_COORDINATION_TIME>	Amended Coordination Time
AMND_R_ALT	4	1	#Y#	AMENDED_DATA	<AMND_R_ALT>230</AMND_R_ALT>	Amended Reported Altitude
AMND_ALT	4	1	#Y#	AMENDED_DATA	<AMND_ALT>200</AMND_ALT>	Amended Filed Altitude

AMND_CURRENT_ROUTE		1	#Y#	AMENDED_DATA	<AMND_CURRENT_ROUTE> MDW/.GIJ292029..CRL.J584.FQM..LVZ..SAX..IS </AMND_CURRENT_ROUTE>	Amended Current Route
--------------------	--	---	-----	--------------	---	-----------------------

Note: #Y# indicates that at least one of these child tags must appear in the parent tag container.

3.3. AZ Message

Tag	Lv	#	Rq?	Parent	Example	Description
MSG	2	1	Y	ASDI_DATA	<MSG>data</MSG>	Container for message
HEADER	3	1	Y	MSG	<HEADER>Header data</HEADER>	Container for header data
SEQ	4	1	Y	HEADER	<SEQ>0012</SEQ>	Sequence number of message. Valid values are 0000 – FFFF (Hex)
TIMESTAMP	4	1	Y	HEADER	<TIMESTAMP>10134248</TIMESTAMP>	Timestamp when AF message was sent by host (ddhhmmss)
SRC	4	1	Y	HEADER	<SRC>KZAU</SRC>	Center or Tracon in which AF generated
TYPE	3	1	Y	MSG	<TYPE>AZ</TYPE>	Type of message - AZ
ID	3	1	Y	MSG	<ID>Flight ID data</ID>	Container for Flight ID data
ACID	4	1	Y	ID	<ACID>SWA917</ACID>	Flight call sign
CID	4	1	N	ID	<CID>123</CID>	Computer ID of flight
ORIG	3	1	Y	MSG	<ORIG>MDW</ORIG>	Origin
DEST	3	1	Y	MSG	<DEST>ISP</DEST>	Destination
ETA	3	1	Y	MSG	<ETA>P1413</ETA>	Arrival Time

3.4. BZ Message

Tag	L v	#	Rq ?	Parent	Example	Description
MSG	2	1	Y	ASDI_DATA	<MSG>data</MSG>	Container for message
HEADER	3	1	Y	MSG	<HEADER>Header data</HEADER>	Container for header data
SEQ	4	1	Y	HEADER	<SEQ>0012</SEQ>	Sequence number of message. Valid values are 0000 – FFFF (Hex)
TIMESTAMP	4	1	Y	HEADER	<TIMESTAMP>10134248</TIMESTAMP>	Timestamp when BZ message was sent by host (ddhhmmss)
SRC	4	1	Y	HEADER	<SRC>KZAU</SRC>	Center or Tracon in which BZ generated
TYPE	3	1	Y	MSG	<TYPE>BZ</BZ>	Type of message - BZ
ID	3	1	Y	MSG	<ID>Flight ID data</ID>	Container for Flight ID data
ACID	4	1	Y	ID	<ACID>SWA917</ACID>	Flight call sign
CID	4	1	N	ID	<CID>123</CID>	Computer ID of flight
ORIG	4	1	Y	MSG	<ORIG>MDW</ORIG>	Origin
DEST	4	1	Y	MSG	<DEST>ISP</DEST>	Destination
BEACON_CODE	4	1	Y	MSG	<BEACON_CODE>1413</BEACON_CODE>	Beacon code (octal only)

3.5. DZ Message

Tag	L v	#	Rq ?	Parent	Example	Description
MSG	2	1	Y	ASDI_DATA	<MSG>data</MSG>	Container for message
HEADER	3	1	Y	MSG	<HEADER>Header data</HEADER>	Container for header data
SEQ	4	1	Y	HEADER	<SEQ>0012</SEQ>	Sequence number of message. Valid values are 0000 – FFFF (Hex)
TIMESTAMP	4	1	Y	HEADER	<TIMESTAMP>10134248</TIMESTAMP>	Timestamp when DZ message was sent by host (ddhhmmss)
SRC	4	1	Y	HEADER	<SRC>KZAU</SRC>	Center or Tracon in which DZ generated
TYPE	3	1	Y	MSG	<TYPE>DZ</DZ>	Type of message - DZ
ID	3	1	Y	MSG	<ID>Flight ID data</ID>	Container for Flight ID data
ACID	4	1	Y	ID	<ACID>SWA917</ACID>	Flight call sign
CID	4	1	N	ID	<CID>123</CID>	Computer ID of flight
AIRCRAFT_DATA	3	1	Y	MSG	<AIRCRAFT_DATA>Acft data</AIRCRAFT_DATA>	Container for Aircraft Data
NUM_ACFT	4	1	N	AIRCRAFT_DATA	<NUM_ACFT>2</NUM_ACFT>	Number of aircraft
SPECIAL_ACFT	4	1	N	AIRCRAFT_DATA	<SPECIAL_ACFT>L</SPECIAL_ACFT>	Special Aircraft Qualifier
EQUIP_TYPE	4	1	Y	AIRCRAFT_DATA	<EQUIP_TYPE>A319</EQUIP_TYPE>	Aircraft Type
AIRBORNE_EQUIP_QUAL	4	1	N	AIRCRAFT_DATA	<AIRBORNE_EQUIP_QUAL>Q</AIRBORNE_EQUIP_QUAL>	Airborne Equipment Qualifier
ORIG	3	1	Y	MSG	<ORIG>MDW</ORIG>	Origin
DEST	3	1	Y	MSG	<DEST>ISP</DEST>	Destination
ETD	3	1	Y	MSG	<ETD>1112</ETD>	Actual Departure Time
ETA	3	1	Y	MSG	<ETA>P1413</ETA>	Estimated Arrival Time

3.6. FZ Message

Tag	L v	#	Rq ?	Parent	Example	Description
MSG	2	1	Y	ASDI_DATA	<MSG>data</MSG>	Container for message
HEADER	3	1	Y	MSG	<HEADER>Header data</HEADER>	Container for header data
SEQ	4	1	Y	HEADER	<SEQ>0012</SEQ>	Sequence number of message. Valid values are 0000 – FFFF (Hex)
TIMESTAMP	4	1	Y	HEADER	<TIMESTAMP>10134248</TIMESTAMP>	Timestamp when FZ message was sent by host (ddhhmmss)
SRC	4	1	Y	HEADER	<SRC>KZAU</SRC>	Center or Tracon in which FZ generated
TYPE	3	1	Y	MSG	<TYPE>FZ</TYPE>	Type of message – FZ
ID	3	1	Y	MSG	<ID>Flight ID data</ID>	Container for Flight ID data
ACID	4	1	Y	ID	<ACID>SWA917</ACID>	Flight call sign
CID	4	1	N	ID	<CID>123</CID>	Computer ID of flight
AIRCRAFT_DATA	3	1	Y	MSG	<AIRCRAFT_DATA>Acft data</AIRCRAFT_DATA>	Container for Aircraft Data
NUM_ACFT	4	1	N	AIRCRAFT_DATA	<NUM_ACFT>2</NUM_ACFT>	Number of aircraft
SPECIAL_ACFT	4	1	N	AIRCRAFT_DATA	<SPECIAL_ACFT>L</SPECIAL_ACFT>	Special Aircraft Qualifier
EQUIP_TYPE	4	1	Y	AIRCRAFT_DATA	<EQUIP_TYPE>A319</EQUIP_TYPE>	Aircraft Type
AIRBORNE_EQUIP_QUAL	4	1	N	AIRCRAFT_DATA	<AIRBORNE_EQUIP_QUAL>Q</AIRBORNE_EQUIP_QUAL>	Airborne Equipment Qualifier
SPEED	3	1	Y	MSG	<SPEED>200</SPEED>	Air Speed
COORDINATION_FIX	3	1	Y	MSG	<COORDINATION_FIX>GOPEV</COORDINATION_FIX>	Coordination Fix
COORDINATION_TIME	3	1	Y	MSG	<COORDINATION_TIME>P1410</COORDINATION_TIME>	Coordination Time
R_ALT	3	1	*Y*	MSG	<R_ALT>230</R_ALT>	Reported Altitude
ALT	3	1	*Y*	MSG	<ALT>200</ALT>	Filed Altitude
CURRENT_ROUTE	3	1	Y	MSG	<CURRENT_ROUTE>MDW/.GIJ29029..CRL.J584.FQM..LVZ..SAX..ISP</CURRENT_ROUTE>	Current Route

Note: *Y* indicates that one and only one of these tags must be in a RZ message.

3.7. RT Message

Tag	L v	#	Rq ?	Parent	Example	Description
MSG	2	1	Y	ASDI_DATA	<MSG>data</MSG>	Container for message
HEADER	3	1	Y	MSG	<HEADER>Header data</HEADER>	Container for header data
SEQ	4	1	Y	HEADER	<SEQ>0012</SEQ>	Sequence number of message. Valid values are 0000 – FFFF (Hex)
TIMESTAMP	4	1	Y	HEADER	<TIMESTAMP>10134248</TIMESTAMP>	Timestamp when DZ message was sent by host (ddhhmmss)
SRC	4	1	Y	HEADER	<SRC>ETMS</SRC>	Center or Tracon in which DZ generated
TYPE	3	1	Y	MSG	<TYPE>RZ</TYPE>	Type of message – RT
ID	3	1	Y	MSG	<ID>Flight ID data</ID>	Container for Flight ID data
ACID	4	1	Y	ID	<ACID>SWA917</ACID>	Flight call sign
CID	4	1	N	ID	<CID>123</CID>	Computer ID of flight (proposed flights only)
ORIG	3	1	Y	MSG	<ORIG>MDW</ORIG>	Origin
DEST	3	1	Y	MSG	<DEST>ISP</DEST>	Destination
STATUS	3	1	Y	MSG	<STATUS>A</STATUS>	Status of flight
AC_CAT	3	1	Y	MSG	<AC_CAT>J</AC_CAT>	Aircraft Category
USR_CAT	3	1	Y	MSG	<USR_CAT>C</USR_CAT>	User Category
ETD	3	1	N	MSG	<ETD>1443</ETD>	Estimated Departure Time
ETA	3	1	N	MSG	<ETA>1623</ETA>	Estimated Arrival Time
ARR_FIX_TIME	3	1	N	MSG	<ARR_FIX_TIME>1608</ARR_FIX_TIME>	Arrival Fix Time
OGTD	3	1	N	MSG	<OGTD>1439</OGTD>	Original Departure Time
OGTA	3	1	N	MSG	<OGTA>1607</OGTA>	Original Arrival Time
CTD	3	1	N	MSG	<CTD>1500</CTD>	Controlled Departure Time
CTA	3	1	N	MSG	<CTA>1645</CTA>	Controlled Arrival Time
DCENTR	3	1	Y	MSG	<DCENTR>G</DCENTR>	Departure Center
GENERATED_BY	3	1	Y	MSG	<GENERATED_BY>FZ</GENERATED_BY>	Message type that initiated creation of RT message
WAYPOINT_LIST	3	1	N	MSG	<WAYPOINT_LIST>4147N/08745W,4147N/08723W, 4203N/08327W,4157N,08216W,4155N/08150W, 4152N,08115W,4148N,08035W,4140N,07912W, 4132N/07811W,4131N/07758W,4120N/07647W, 4116N/07541W,4104N/07432W,4048N/07306W </WAYPOINT_LIST>	List of Waypoints traversed
SECTOR_LIST	3	1	N	MSG	<SECTOR_LIST>ZAUORD,ZAU81,ZAU82 </SECTOR_LIST>	List of Sectors traversed

FIX_LIST	3	1	N	MSG	<FIX_LIST>GIJ,CRL,BUYKK,KEEHO,FAILS,BEELR, DORET,ZORBO,BRIAR,SLT,FQM,LVZ,SAX,DPK </FIX_LIST>	List of Fixes traversed
AIRWAY_LIST	3	1	N	MSG	<AIRWAY_LIST>V10,V6,J584,J554,J190 </AIRWAY_LIST>	List of Airways traversed
CENTER_LIST	3	1	N	MSG	<CENTER_LIST>G,C,N,B,N</CENTER_LIST>	List of Centers traversed
CURRENT_ROUTE	3	1	N	MSG	<CURRENT_ROUTE> MDW./.GIJ292029..CRL.J584.FQM..LVZ..SAX..ISP /1611 </CURRENT_ROUTE>	Current Route

3.8. RZ Message

Tag	L v	#	Rq ?	Parent	Example	Description
MSG	2	1	Y	ASDI_DATA	<MSG>data</MSG>	Container for message
HEADER	3	1	Y	MSG	<HEADER>Header data</HEADER>	Container for header data
SEQ	4	1	Y	HEADER	<SEQ>0012</SEQ>	Sequence number of message. Valid values are 0000 – FFFF (Hex)
TIMESTAMP	4	1	Y	HEADER	<TIMESTAMP>10134248</TIMESTAMP>	Timestamp when DZ message was sent by host (ddhhmmss)
SRC	4	1	Y	HEADER	<SRC>KZAU</SRC>	Center or Tracon in which DZ generated
TYPE	3	1	Y	MSG	<TYPE>RZ</TYPE>	Type of message – RZ
ID	3	1	Y	MSG	<ID>Flight ID data</ID>	Container for Flight ID data
ACID	4	1	Y	ID	<ACID>SWA917</ACID>	Flight call sign
CID	4	1	N	ID	<CID>123</CID>	Computer ID of flight (proposed flights only)
ORIG	3	1	Y	MSG	<ORIG>MDW</ORIG>	Origin
DEST	3	1	Y	MSG	<DEST>ISP</DEST>	Destination

3.9. TO Message

Tag	L v	#	Rq ?	Parent	Example	Description
MSG	2	1	N	ASDI_DATA	<MSG>data</MSG>	Container for message
HEADER	3	1	Y	MSG	<HEADER>Header data</HEADER>	Container for header data
SEQ	4	1	Y	HEADER	<SEQ>0012</SEQ>	Sequence number of message. Valid values are 0000 – FFFF (Hex)
TIMESTAMP	4	1	Y	HEADER	<TIMESTAMP>10134248</TIMESTAMP>	Timestamp when UZ message was sent by host (ddhhmmss)
SRC	4	1	Y	HEADER	<SRC>ETMS</SRC>	Center or Tracon in which UZ generated
TYPE	3	1	Y	MSG	<TYPE>TO</TZ>	Type of message - TO
ID	3	1	Y	MSG	<ID>Flight ID data</ID>	Container for Flight ID data
ACID	4	1	Y	ID	<ACID>SWA917</ACID>	Flight call sign
CID	4	1	Y	ID	<CID>123</CID>	Computer ID of flight
SPEED	3	1	Y	MSG	<SPEED>200</SPEED>	Air Speed
POS_REPORTED	3	1	Y	MSG	<POS_REPORTED>Reported Position data</POS_REPORTED>	Container for Reported Position data
RPT_TIME	4	1	Y	POS_REPORTED	<RPT_TIME>13/1456</RPT_TIME>	Reported position time (dd/hhmm)
RPT_R_ALT	4	1	Y	POS_REPORTED	<RPT_R_ALT>360</RPT_R_ALT>	Reported position altitude
RPT_POS	4	1	Y	POS_REPORTED	<RPT_POS>5900N/0400W</RPT_POS>	Reported position lat/lon
POS_PLANNED	3	1	Y	MSG	<POS_PLANNED>Planned Position data</POS_PLANNED>	Container for Planned Position data
PLN_NUM	4	2*	-Y-	POS_PLANNED	<PLN_NUM>1</PLN_NUM>	Sequence number of Planned Position container (optional, can have between 0 – 2 in TO message)
PLN_TIME	4	1	-Y-	POS_PLANNED	<PLN_TIME>13/1556</PLN_TIME>	Reported position time (dd/hhmm)
PLN_R_ALT	4	1	-Y-	POS_PLANNED	<PLN_R_ALT>360</PLN_R_ALT>	Planned position altitude
PLN_POS	4	1	-Y-	POS_PLANNED	<PLN_POS>5900N/0400W</PLN_POS>	Planned position lat/lon

Note: -Y- indicates that these tags must be in a TO message if their parent tag is in the message.

3.10. TZ Message

Tag	L v	#	Rq ?	Parent	Example	Description
MSG	2	1	Y	ASDI_DATA	<MSG>data</MSG>	Container for message
HEADER	3	1	Y	MSG	<HEADER>Header data</HEADER>	Container for header data
SEQ	4	1	Y	HEADER	<SEQ>0012</SEQ>	Sequence number of message. Valid values are 0000 – FFFF (Hex)
TIMESTAMP	4	1	Y	HEADER	<TIMESTAMP>10134248</TIMESTAMP>	Timestamp when TZ message was sent by host (ddhhmmss)
SRC	4	1	Y	HEADER	<SRC>KZAU</SRC>	Center or Tracon in which TZ generated
TYPE	3	1	Y	MSG	<TYPE>TZ</TZ>	Type of message – TZ
ID	3	1	Y	MSG	<ID>Flight ID data</ID>	Container for Flight ID data
ACID	4	1	Y	ID	<ACID>DLH436</ACID>	Flight call sign
CID	4	1	N	ID	<CID>123</CID>	Computer ID of flight
G_SPEED	3	1	Y	MSG	<G_SPEED>401</G_SPEED>	Ground Speed
R_ALT	3	1	Y	MSG	<R_ALT>380</R_ALT>	Reported Altitude
POSITION	3	1	Y	MSG	<POSITION>5137N/06859W</POSITION>	Reported position (Lat/Lon – degrees/minutes)

3.11. UZ Message

Tag	L v	#	Rq ?	Parent	Example	Description
MSG	2	1	Y	ASDI_DATA	<MSG>data</MSG>	Container for message
HEADER	3	1	Y	MSG	<HEADER>Header data</HEADER>	Container for header data
SEQ	4	1	Y	HEADER	<SEQ>0012</SEQ>	Sequence number of message. Valid values are 0000 – FFFF (Hex)
TIMESTAMP	4	1	Y	HEADER	<TIMESTAMP>10134248</TIMESTAMP>	Timestamp when UZ message was sent by host (ddhhmmss)
SRC	4	1	Y	HEADER	<SRC>KZAU</SRC>	Center or Tracon in which UZ generated
TYPE	3	1	Y	MSG	<TYPE>UZ</TZ>	Type of message - UZ
ID	3	1	Y	MSG	<ID>Flight ID data</ID>	Container for Flight ID data
ACID	4	1	Y	ID	<ACID>SWA917</ACID>	Flight call sign
CID	4	1	Y	ID	<CID>123</CID>	Computer ID of flight
AIRCRAFT_DATA	3	1	Y	MSG	<AIRCRAFT_DATA>Acft data</AIRCRAFT_DATA>	Container for Aircraft Data
NUM_ACFT	4	1	N	AIRCRAFT_DATA	<NUM_ACFT>2</NUM_ACFT>	Number of aircraft
SPECIAL_ACFT	4	1	N	AIRCRAFT_DATA	<SPECIAL_ACFT>L<SPECIAL_ACFT>	Special Aircraft Qualifier
EQUIP_TYPE	4	1	Y	AIRCRAFT_DATA	<EQUIP_TYPE>A319</EQUIP_TYPE>	Aircraft Type
AIRBORNE_EQUIP_QUAL	4	1	N	AIRCRAFT_DATA	<AIRBORNE_EQUIP_QUAL>Q</AIRBORNE_EQUIP_QUAL>	Airborne Equipment Qualifier
SPEED	3	1	Y	MSG	<SPEED>200</SPEED>	Air Speed
BOUNDAY_POSITION	3	1	Y	MSG	<BDRY_POS>4029N/09347W</BDRY_POS>	Boundary position (lat/lon)
BOUNDARY_TIME	3	1	Y	MSG	<BDRY_TIME>e1410</BDRY_TIME>	Boundary Time
R_ALT	3	1	N	MSG	<R_ALT>230</R_ALT>	Reported Altitude
CURRENT_ROUTE	3	1	Y	MSG	<CURRENT_ROUTE>MDW/.GIJ292029..CRL.J584.FQM..LVZ..SAX..ISP</CURRENT_ROUTE>	Current Route

Appendix A: Sample XML Data

The following XML example includes a <MSG> container for each message type that is in the current ASDI feed.

```
<?xml version="1.0"?>
<ASDI_DATA>
  <MSG>
    <HEADER>
      <SEQ>0000</SEQ>
      <TIMESTAMP>13145945</TIMESTAMP>
      <SRC>KZHN</SRC>
    </HEADER>
    <TYPE>TZ</TYPE>
    <ID>
      <ACID>AIP392</ACID>
      <CID>466</CID>
    </ID>
    <G_SPEED>120</G_SPEED>
    <R_ALT>010</R_ALT>
    <POSITION>2116N/15757W</POSITION>
  </MSG>
  <MSG>
    <HEADER>
      <SEQ>0001</SEQ>
      <TIMESTAMP>13145939</TIMESTAMP>
      <SRC>CCZW</SRC>
    </HEADER>
    <TYPE>DZ</TYPE>
    <ID>
      <ACID>PAG203</ACID>
    </ID>
    <AIRCRAFT_DATA>
      <SPECIAL_ACFT>L</SPECIAL_ACFT>
      <EQUIP_TYPE>SW4</EQUIP_TYPE>
      <AIRBORNE_EQUIP_QUAL>G</AIRBORNE_EQUIP_QUAL>
    </AIRCRAFT_DATA>
    <ORIG>CYWG</ORIG>
    <DEST>CYOH</DEST>
    <ETD>D1459</ETD>
    <ETA>1550</ETA>
  </MSG>
  <MSG>
    <HEADER>
      <SEQ>0002</SEQ>
      <TIMESTAMP>13145944</TIMESTAMP>
      <SRC>KZDC</SRC>
    </HEADER>
    <TYPE>UZ</TYPE>
    <ID>
      <ACID>USA418</ACID>
    </ID>
    <AIRCRAFT_DATA>
```

```

        <SPECIAL_ACFT>T</SPECIAL_ACFT>
        <EQUIP_TYPE>B734</EQUIP_TYPE>
        <AIRBORNE_EQUIP_QUAL>J</AIRBORNE_EQUIP_QUAL>
    </AIRCRAFT_DATA>
    <SPEED>0426</SPEED>
    <BOUNDARY_CROSSING_POS>3528N/07948W</BOUNDARY_CROSSING_POS>
    <BOUNDARY_CROSSING_TIME>E1503</BOUNDARY_CROSSING_TIME>
    <ALT>290</ALT>

<CURRENT_ROUTE>CLT.PAN6.MERIL..RDU.J52.RIC.OTT6.BWI/1548</CURRENT_ROUTE>
</MSG>
<MSG>
    <HEADER>
        <SEQ>0003</SEQ>
        <TIMESTAMP>13145950</TIMESTAMP>
        <SRC>KZDC</SRC>
    </HEADER>
    <TYPE>AF</TYPE>
    <ID>
        <ACID>N541RS</ACID>
    </ID>
    <ORIG>RWI</ORIG>
    <DEST>IAD</DEST>
    <AMENDED_DATA>
        <AMND_COORDINATION_FIX>3559N/07742W</AMND_COORDINATION_FIX>
        <AMND_COORDINATION_TIME>E1500</AMND_COORDINATION_TIME>

<AMND_CURRENT_ROUTE>RWI./.TYI237008..TYI..FAK.BARIN1.IAD</AMND_CURRENT_ROUTE>
    </AMENDED_DATA>
</MSG>
<MSG>
    <HEADER>
        <SEQ>0004</SEQ>
        <TIMESTAMP>13150000</TIMESTAMP>
        <SRC>KZDC</SRC>
    </HEADER>
    <TYPE>BZ</TYPE>
    <ID>
        <ACID>ACA908</ACID>
        <CID>171</CID>
    </ID>
    <ORIG>YYZ</ORIG>
    <DEST>FLL</DEST>
    <BEACON_CODE>2223</BEACON_CODE>
</MSG>
<MSG>
    <HEADER>
        <SEQ>0005</SEQ>
        <TIMESTAMP>13145952</TIMESTAMP>
        <SRC>ETMS</SRC>
    </HEADER>
    <TYPE>RT</TYPE>
    <ID>

```

```

<ACID>SWA917</ACID>
<CID>648</CID>
</ID>

<ETD>1443</ETD>
<ETA>1623</ETA>
<ARR_FIX_TIME>1608</ARR_FIX_TIME>
<STATUS>A</STATUS>
<AC_CAT>J</AC_CAT>
<USER_CAT>C</USER_CAT>
<OGTD>1439</OGTD>
<OGTA>1607</OGTA>
<ORIG>MDW</ORIG>
<DEST>ISP</DEST>
<DCENTR>G</DCENTR>
<GENERATED_BY>UZ</GENERATED_BY>

<WAYPOINT_LIST>4147N/08745W,4147N/08723W,4203N/08327W,4157N/08216W,415
5N/08150W,4152N/08115W,4148N/08035W,4140N/07912W,4132N/07811W,4131N/07
758W,4120N/07647W,4116N/07541W,4104N/07432W,4048N/07306W</WAYPOINT_LIST>
<SECTOR_LIST>ZAUORD,ZAU81,ZAU80,ZAU82</SECTOR_LIST>

<FIX_LIST>GIJ,CRL,BUYKK,KEEHO,FAILS,BEELR,DORET,ZORBO,BRIAR,SLT,FQM,LV
Z,SAX,DPK,</FIX_LIST>
<AIRWAY_LIST>V10,V6,J584,J554,J190</AIRWAY_LIST>
<CENTER_LIST>G,C,N,B,N</CENTER_LIST>

<CURRENT_ROUTE>MDW./.GIJ292029..CRL.J584.FQM..LVZ..SAX..ISP/1611</CURRENT_ROUTE>
</MSG>
<MSG>
<HEADER>
<SEQ>0006</SEQ>
<TIMESTAMP>13145945</TIMESTAMP>
<SRC>KZLA</SRC>
</HEADER>
<TYPE>AZ</TYPE>
<ID>
<ACID>SWA2945</ACID>
</ID>
<ORIG>PHX</ORIG>
<DEST>LAS</DEST>
<ETA>E1451</ETA>
</MSG>
<MSG>
<HEADER>
<SEQ>0007</SEQ>
<TIMESTAMP>13150006</TIMESTAMP>
<SRC>KZID</SRC>
</HEADER>
<TYPE>FZ</TYPE>
<ID>
<ACID>N770CH</ACID>
<CID>262</CID>

```

```

</ID>
<AIRCRAFT_DATA>
  <EQUIP_TYPE>LJ31</EQUIP_TYPE>
  <AIRBORNE_EQUIP_QUAL>Q</AIRBORNE_EQUIP_QUAL>
</AIRCRAFT_DATA>
<SPEED>0440</SPEED>
<COORDINATION_FIX>OSU</COORDINATION_FIX>
<COORDINATION_TIME>P1700</COORDINATION_TIME>
<ALT>410</ALT>

<CURRENT_ROUTE>OSU.J186.BULEY..SPA..OMN..BCT/0210</CURRENT_ROUTE>
</MSG>
<MSG>
  <HEADER>
    <SEQ>0008</SEQ>
    <TIMESTAMP>13150008</TIMESTAMP>
    <SRC>KZID</SRC>
  </HEADER>
  <TYPE>RZ</TYPE>
  <ID>
    <ACID>N37BM</ACID>
    <CID>625</CID>
  </ID>
  <ORIG>JVY</ORIG>
  <DEST>CAK</DEST>
</MSG>
<MSG>
  <HEADER>
    <SEQ>0009</SEQ>
    <TIMESTAMP>13150007</TIMESTAMP>
    <SRC>ETMS</SRC>
  </HEADER>
  <TYPE>TO</TYPE>
  <ID>
    <ACID>N614AF</ACID>
  </ID>
  <SPEED>407</SPEED>
  <POS_REPORTED>
    <RPT_TIME>13/1456</RPT_TIME>
    <RPT_R_ALT>360</RPT_R_ALT>
    <RPT_POS>5900N/04000W</RPT_POS>
  </POS_REPORTED>
  <POS_PLANNED>
    <PLN_NUMBER>1</PLN_NUMBER>
    <PLN_TIME>13/1546</PLN_TIME>
    <PLN_ALT>360</PLN_ALT>
    <PLN_POS>5700N/05000W</PLN_POS>
  </POS_PLANNED>
  <POS_PLANNED>
    <PLN_NUMBER>2</PLN_NUMBER>
    <PLN_TIME>00/0000</PLN_TIME>
    <PLN_ALT>360</PLN_ALT>
    <PLN_POS>5531N/05701W</PLN_POS>
  </POS_PLANNED>
</MSG>

```

</ASDI_DATA>

Appendix B: Issues Concerning Compression

As explained in Section 2.1, compression is needed because the XML version of the feed, if uncompressed, would require an unacceptably large bandwidth. The proposal is that the ASDI Server will hold messages until either m messages have arrived or s seconds have elapsed, at which time it will compress and send the accumulated messages, where m and s are configurable parameters.

This leads to the question of what values should be chosen for m and s , and vendor input is desired on this question. Table B-1 shows the implications of choosing different values of m . The values in this table are based on a preliminary version of the XML version of the ASDI Server; it should not be expected that these values are exact, but they are close enough to evaluate the options.

The interpretation of the columns in this table is as follows. Each row shows the results for a value of m , i.e., the number of messages that are batched together to be compressed. To see how to interpret this table, consider the fifth row. Each column has the following interpretation.

- The first column shows that the value chosen for m is 16.
- The second column shows that these 16 messages in the current, flat version of the ASDI feed take up 763 bytes.
- The third column shows that if these 16 messages are converted to XML using the format described in this document, then they take up 3592 bytes.
- The fourth column shows that when these 16 XML messages are compressed, they take up 628 bytes.
- The fifth column shows that the percentage size decrease in going from the uncompressed XML to the compressed XML, i.e., going from 3592 to 628 bytes, is 82.52 percent.
- The sixth column shows that the size change in going from the current, flat data to the compressed data, i.e., going from 763 to 628 bytes, is a decrease of 17.7 percent.

Table B-1: Results of Experimentation with Different Values of m

Value Chosen For m	Flat (bytes)	XML (bytes)	XML Comp (bytes)	XML->XML Comp (% size decrease)	Flat -> XML Comp (% size increase) or (-% size decrease)
1	52	281	237	15.66	355.77
2	105	516	277	46.32	163.81
4	204	969	346	64.30	69.61
8	390	1843	443	75.97	11.97
16	763	3592	628	82.52	-17.70
32	1524	7228	1120	84.51	-26.51
64	3260	15118	2155	85.75	-33.90
128	7081	31567	4556	85.57	-35.66
256	19261	74771	12452	83.35	-35.36
512	32879	136155	19669	85.56	-40.18

If one compares the second and third columns, then, as expected, one sees that uncompressed XML data takes up many more bytes than the current data. The interesting comparison, however, is to compare the second and sixth columns. For small values of m , the compressed XML files are much larger than the flat files; this is because the uncompressed XML files are larger than the flat files but are not large enough to allow the compression algorithm to achieve much compression. For larger values of m , however, the

compression algorithm, because it has more data to work with, can not only greatly compress the XML messages but also make them smaller than the flat messages.

The question is: **What values should be selected for m and s ?** To avoid undue complexity, the same value should be chosen for each vendor. The considerations are:

- The higher the value of m , the less bandwidth is used but the more the delay until a message is received.
- The higher the value of s , the less bandwidth is used but the more the delay until a message is received.

Keep in mind that bandwidth is only an issue during busy periods, and in these periods there will be more than 100 messages per second. This means that even with a fairly high value of m , there will be minimal delay while a message is held to be compressed. If a second is taken to be a delay that is acceptable, then a very high value of m such as $m = 64$ and $s = 1$ would achieve most of the achievable compression with a maximum delay of at most a second, but at the cost of making the transmission lumpy, i.e., no data being sent for half a second or more followed by a burst of data.